AFRL-IF-RS-TR-2003-285
Final Technical Report
December 2003

PROTECTING NETWORK QUALITY OF SERVICE AGAINST DENIAL OF SERVICE ATTACKS

North Carolina State University

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. H544

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2003-285 has been reviewed and is approved for publication

APPROVED: /s/

KEVIN A. KWIAT
Project Engineer

FOR THE DIRECTOR: /s/

WARREN H. DEBANY, JR.
Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
DECEMBER 2003

3. REPORT TYPE AND DATES COVERED
Final Aug 99 – Feb 03

4. TITLE AND SUBTITLE

PROTECTING NETWORK QUALITY OF SERVICE AGAINST DENIAL OF SERVICE ATTACKS

5. FUNDING NUMBERS
C - F30602-99-1-0540
PE - 62301E
PR - H544
TA - 10
WU - 01

6. AUTHOR(S)

Douglas Reeves, Felix Wu, Peter Wurman, Dan Stevenson, and Xiaoyong Wu

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

North Carolina State University
Department of Computer Science
Raleigh North Carolina 27695-8206

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington Virginia 22203-1714

AFRL/IFGA
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2003-285

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Kevin A. Kwiat/IFGA/(315) 330-1692/ Kevin.Kwiat@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*

As quality of Service (QiS) capabilities are added to the Internet, our nation’s business and research infrastructure will increasingly depend on their fault tolerance and survivability. Current frameworks and protocols (such as the Resource ReSer Vation Protocol (RSVP), Integrated Services (IntServ), and Differentiated Services (DiffServ)) that provide quality of service to networks are vulnerable to attacks of abuse and denial of service. Once QoS mechanisms are deployed, they will become tempting targets for malicious hackers or adversaries. It is wise to anticipate such attacks and develop effective and practical defenses prior to widespread deployment.

14. SUBJECT TERMS

Quality of Service, Denial of Service, Fault Tolerance, Survivability, Market Mechanisms

15. NUMBER OF PAGES
43

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

Abstract

As quality of service (*QoS*) capabilities are added to the Internet, our nation's business and research infrastructure will increasingly depend on their fault tolerance and survivability. Current frameworks and protocols (such as the *Resource ReSerVation Protocol (RSVP)*, *Integrated Services (IntServ)*, and *Differentiated Services (DiffServ))* that provide quality of service to networks are vulnerable to attacks of abuse and denial of service. Once QoS mechanisms are deployed, they will become tempting targets for malicious hackers or adversaries. We believe it is wise to anticipate such attacks and develop effective and practical defenses prior to widespread deployment.

In this project we investigated the vulnerabilities of current protocols to attacks on QoS, and we proposed methods of protecting against such attacks. All methods were investigated in depth by simulation, implementation, or a combination of the two. In this document, we summarize our findings and the results we produced. The parts of this project are as follows:

1. Protecting RSVP Against Insider Attacks, Using Selective Digital Signatures with Conflict Detection
2. Preventing Denial of Service Attacks on QoS by Pricing of Network Resources
3. Protection of Reliable Multicast Against Receiver-Based Attacks
4. Capacity Planning for Networks Supporting DiffServ and MPLS
5. Detection of Denial-of-QoS Attacks on DiffServ Networks
6. Statistical Anomaly Detection for TCP Packet Dropping Attacks
7. Automatic Generation Of IPSec/VPN Security Policies In An Intra-domain Environment
8. Tracing Attack Traffic Through Stepping Stones Using Watermarking

All outputs mentioned as part of this project are available on the ARQoS website (http://arqos.csc.ncsu.edu). As part of this project, 9 students (4 PhD, 5 MS with thesis) were or will soon be graduated, 15 conference or workshop papers were presented and published, 2 journal papers were published or are currently in review, and 7 software packages were released.

Acknowledgments

Many students have worked on the ARQoS project. These include:

- Zhi Fu
- Errin Fulp
- Khurram Khan
- Zhaoning Li
- Vinay Mahadik
- Nipul Shah
- Ashish Sureka
- Xinyuan Wang
- Kehang Wu
- Tsung-Li Wu
- Xiaobing Zhang

Their efforts are gratefully acknowledged.

In addition, we wish to acknowledge the support of MCNC and the Department of Computer Science at NC State.

The support of DARPA has been generous. Dr. Douglas Maughan runs a professional and aggressive program with some exceptional researchers; it has been an honor to be included among them. He has been ably assisted by Mike Ferguson and John Drake. Dr. Kevin Kwiat of the Air Force Rome Labs has also been extremely helpful. We thank them all.

Overview

As quality of service (*QoS*) capabilities are added to the Internet, our nation's business and research infrastructure will increasingly depend on their fault tolerance and survivability. Current frameworks and protocols, such as the *Resource ReSerVation Protocol (RSVP)* [RFC2205], *Integrated Services (IntServ)* [RFC1633], and *Differentiated Services (DiffServ)* [RFC2475] that provide quality of service to networks are vulnerable to attacks of abuse and denial of service. Once QoS mechanisms are deployed, they will become tempting targets for malicious hackers or adversaries. We believe it is wise to anticipate such attacks and develop effective and practical defenses prior to widespread deployment.

TCP is the main transport protocol in the Internet for providing reliable delivery of packets. *UDP* is the other important transport protocol; UDP does not guarantee reliable delivery, and is used for applications where occasional packet loss is tolerable. In this document, a *flow* or *connection* refers to a TCP or UDP connection between two applications, over which many packets may be sent. A flow is by definition bi-directional, meaning packets may be sent in both directions.

RSVP is a signaling protocol that was first proposed in the early 1990's. It allows a sending application to declare the characteristics of the traffic it will send out. The receiving application computes the appropriate bandwidth for this traffic, and the QoS that it desires from the network. RSVP allows the receiver to reserve this amount of bandwidth along the path between the sender and the receiver, for the duration of the flow. RSVP is considered to be a very safe, predictable, and conservative means of providing QoS on a per-flow basis. It is also considered to be somewhat *heavy-weight*, or expensive to implement and operate.

DiffServ was proposed in the mid 1990's, and attempts to provide QoS at lower cost than RSVP. DiffServ provides relative or qualitative QoS rather than absolute guarantees of QoS. It provides only a few choices of QoS classes, and it only controls QoS for large aggregations of flows, rather than for individual flows. The basic classes, in order from highest to lowest quality, are *EF (Expedited Forwarding)*, *AF (Assured Forwarding)*, and *BE (Best Effort).* A *Service Level Agreement*, or *SLA*, defines the QoS that is expected for user traffic.

In this project we investigated the vulnerabilities of current protocols to attacks on QoS, and we proposed methods of protecting against such vulnerabilities. All methods were investigated in depth by simulation, implementation, or a combination of the two. In the following 8 chapters we summarize our findings and the results we produced. The chapter contents are as follows:

1. Protecting RSVP Against Insider Attacks, Using Selective Digital Signatures with Conflict Detection
2. Preventing Denial of Service Attacks on QoS by Pricing of Network Resources
3. Protection of Reliable Multicast Against Receiver-Based Attacks
4. Capacity Planning for Networks Supporting DiffServ and MPLS
5. Detection of Denial-of-QoS Attacks on DiffServ Networks
6. Statistical Anomaly Detection for TCP Packet Dropping Attacks
7. Automatic Generation Of IPSec/VPN Security Policies In An Intra-domain Environment
8. Tracing Attack Traffic Through Stepping Stones Using Watermarking

Note: all outputs mentioned as part of this project are available on the ARQoS website (http://arqos.csc.ncsu.edu).

References

[RFC1633] R. Braden, D. Clark and S. Shenker, “Integrated Services in the Internet Architecture: an Overview”, IETF RFC1633, June 1994.

[RFC2205] R. Braden et al., “Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification”, IETF RFC2205, September 1997.

[RFC2475] S. Blake et al., “An Architecture for Differentiated Services”, IETF RFC2475, December 1998.

Chapter 1

Protecting the Resource Reservation Protocol Against Insider Attacks Using Selective Digital Signatures with Conflict Detection

Summary

RSVP is the IETF standard for reserving resources on a path, in order to ensure network QoS. There has not been much attention paid to the security vulnerabilities of RSVP, which may allow denial of service attacks on QoS. In this part of the project we analyzed the vulnerabilities of RSVP and showed they were not adequately addressed by existing proposals. We then described a method of protection that addresses most of these vulnerabilities.

Our method uses conventional digital signatures to protect the contents of RSVP messages. However, unlike previous techniques, our method uses only end-to-end authentication, rather than hop-by-hop authentication, and is therefore cheaper to implement. Since some of the fields in RSVP messages may legitimately be altered in transit, they cannot be digitally signed by the originator. Rather, we detect conflicts in such mutable fields when intermediate routers alter them in undesirable (and illegal) ways. We show that this combination of end-to-end authentication and conflict detection protects against a much wider set of attacks than previous methods.

Introduction

Ensuring that the traffic generated by a user's application receives the quality of service it requires from the network can only be accomplished if there are adequate network resources (bandwidth, buffer space, and queuing priority) available. RSVP [RFC2205] is an IETF standard for reserving network resources for this purpose. With RSVP, the originator of traffic creates a *PATH* message, which is transmitted to the receiver or receivers. This PATH message specifies the characteristics of the traffic that the sender expects to generate; this specification is called the *Tspec* object. The routers that route the PATH message to the receiver(s) must also inform the receiver of the maximum amount of resources available on the path from the sender to the receiver. This information constitutes the *Adspec* object. The receiver(s), upon receiving the PATH message, will determine how much resources the traffic should receive to ensure adequate QoS. This information is added to the *Flowspec* object, and inserted into a second message type, the *RESV* message. The RESV message is sent from the receiver(s) back to the sender along the reverse of the path taken by the PATH message. The routers receiving the RESV message will record the amount of resources being reserved for this flow, and propagate the RESV message back towards the sender. When the RESV message reaches the sender, resources have been reserved all along the path and transmission of user traffic with the desired QoS may begin.

Periodically (every 30 seconds) this process is repeated to maintain the reservation state in each of the routers. RESV messages from multiple receivers (of traffic from a single sender) are merged if their reservations share some part of the same path. This occurs when the sender's traffic is multicast to a set of receivers. The merging requires that the maximum reservation request of the merged RESV messages is propagated towards the sender. A figure illustrating RSVP operation is shown below.

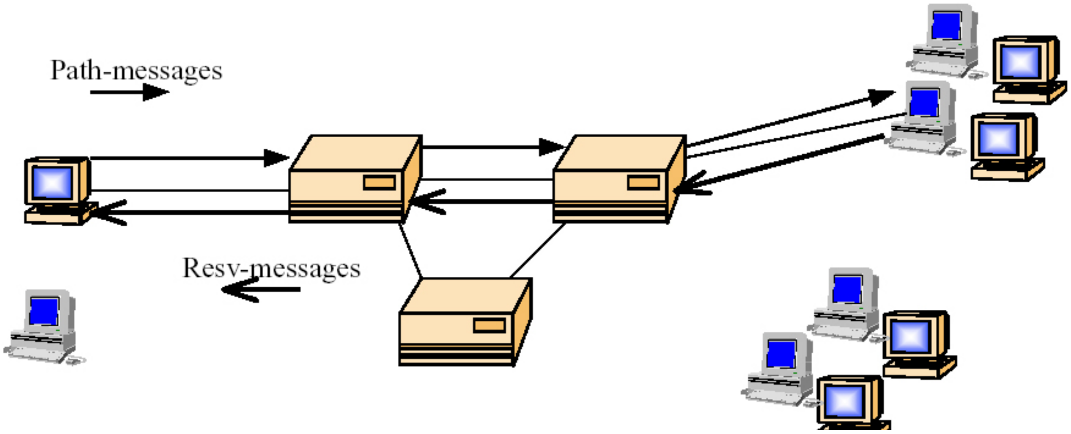


Figure 1. RSVP Operation

If during normal operation of RSVP a reservation should be terminated (it is no longer needed, or a router no longer has sufficient resources to maintain the reservation), path and reservation *TEARDOWN* messages will be generated. Obviously, only routers participating in the reservation should initiate TEARDOWN messages.

Attackers who wish to interfere with the QoS provided by the network may launch two types of attacks. In the first type, the attacker attempts to disrupt the correct operation of RSVP. In the second type of attack, the attacker attempts to interfere with the user traffic which is receiving the guaranteed QoS. In this part of the project, we restricted our attention to the first type of attack, *i.e.,* how to prevent attacks on the correct operation of RSVP (see chapter 6 for more about the second category of attack). Attacks in this first category include:

1. Intercepting and dropping (failing to propagate) PATH and/or RESV messages
2. Illegal modification of traffic requirements (TSpec object), available resource information (Adspec object), or reservation requests (Flowspec object)
3. Forging of TEARDOWN messages by non-participating routers.

We assume in this work that the senders operate in a benign and proper way, and that they will not make false requests or declarations (see chapter 2 for more on this problem). We are concerned with the possibility that routers or receivers may delete, illegally modify, or forge PATH and RESV messages between the sender and receiver(s). To prevent message forgeries, we assume that PATH and RESV messages are digitally signed by the hosts that generate them. To detect message deletion, we assume that failure to create a reservation is detected and the user or a system administrator will be notified, in which case failure diagnosis can be performed. Therefore, we are concerned mainly with illegal modification of RESV messages, and illegal generation of TEARDOWN messages.

The challenges in verifying correct RSVP operation and detecting malicious behavior are the following:

- Message contents can be (legitimately) modified in transit. The Adspec object can be updated by each router on the path. Any router modifying the Flowspec may decrease the amount of available bandwidth, but never increase it. The Flowspec objects from multiple receivers can be merged at the router where two RESV messages join together to follow the same path to the sender. The larger of the two Flowspecs being merged should be forwarded, not the smaller.

- There is a logical relationship between the Tspec object, the Adspec object, and the Flowspec object. The Adspec should not be larger than the Tspec object, and the Flowspec should be not larger than the Adspec object.

Previous work on protecting RSVP has not adequately addressed these issues. Baker et al. [BLT98] have proposed that hop-by-hop authentication be used to detect interference with or forgery of RSVP messages between RSVP-enabled routers. Unfortunately, their approach does not protect against malicious behavior by such RSVP-enabled routers themselves. We describe below a more robust method for protecting RSVP.

Methods, Assumptions, Procedures

Our approach only requires that the sender and receiver digitally sign information in the PATH and RESV messages. Routers on the path between them may need to verify these signatures, but do not themselves have to sign any of the information in these messages. Our modifications are described below:

1. The sender digitally signs those parts of the PATH message which will not be changed in transit, and then sends the PATH message.
2. An RSVP-enabled router on the path between the sender and receiver processes this PATH message, possibly modifies the Adspec object, and forwards it towards the receiver.
3. The receiver, upon receiving this PATH message, verifies the integrity of the unmodified contents of the PATH message.
4. The receiver generates a RESV message, based upon the received PATH message. The receiver signs this message and separately signs the Flowspec object in this message. In addition, the receiver signs the Adspec object which was received in the PATH message, and adds this ("piggybacks it") to the RESV message.
5. An RSVP-enabled router on the path between the receiver and sender processes this RESV message, and may merge multiple RESV messages from multiple receivers. It also verifies that the signed Adspec object in the RESV message is no larger than the value it generated and forwarded itself in the preceding PATH message. If it *is* larger, then some router or the receiver illegally modified the Adspec object.
6. When the RESV message arrives at the sender, the integrity of the contents is verified.
7. The Flowspec object received in this RESV message is signed by the sender, and included in the *next* PATH message generated by this sender (*i.e.*, in the next 30 seconds).
8. An RSVP-enabled router receiving this new PATH message compares the signed Flowspec object with the Flowspec object it forwarded to the sender. If the signed object is smaller than the stored one, the Flowspec object was illegally merged (the smaller of two Flowspecs was chosen, rather than the larger).

Forgery of PATH, RESV, or TEARDOWN messages by routers not part of the path between the sender and the receiver may be detected by requiring all messages to be signed by the originator.

Results and Discussion

A considerably larger number of attacks on RSVP can be detected by this method vs. the method of Baker et al. Specifically, illegal modifications by RSVP-enabled routers on the path from the sender to the receiver, or by the receiver, will be detected. Part of the contribution of this work has been the identification of these additional vulnerabilities that were not previously addressed.

In addition, our method has lower overhead, since signing of message contents is only performed by the end points, *not* by each hop (RSVP-enabled router on the path). End-to-end delay and throughput of

RSVP should be improved with this approach, while at the same time security and robustness have been improved.

Some outcomes of this work are the following:

- This work was presented at the International Workshop on QoS (IWQOS) in 1999, and published as: T-L Wu, S. F. Wu, Z. Fu, H. Huang, and F. Gong, "Securing QoS: Threats to RSVP Messages and Their Countermeasures", *Proc of the Intl. Workshop on QoS (IWQOS '99),* 1999.

This work was part of the PhD theses of Tsung-Li Wu and Zhi Fu. Dr. Wu is now a faculty member at National Tsing Hua University in Taiwan. Dr. Fu is a researcher at Motorola Labs in Chicago.

Conclusions, Recommendations

As protocols for providing QoS become more widespread, it can be expected they will become the target of attacks. RSVP is an example of a signaling protocol that seems difficult to adequately protect, because the message contents are modified along the path to the destination. Conventional authentication signs message contents when they are generated and verifies integrity when they are received. This approach cannot be used when message contents can be (legitimately) changed in transit. The approach we advocate instead is (a) recording the value transmitted (b) signing the value received (c) reflecting this received value back to the senders, and (d) comparing the received value with the transmitted value to detect invalid modifications. This may be thought of as a blending of intrusion detection and conventional authentication. We advocate the standardization and adoption of this approach to protect RSVP from potential attacks by both "outsiders" (non-trusted routers) and "insiders" (trusted routers + senders and receivers).

References

[RFC2205] R. Braden et al., "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", IETF RFC2205, September 1997.

[BLT98] F. Baker, B. Lindall, and M. Talwar. RSVP Cryptographic Authentication. Internet Draft, IETF, November 1998. Network Working Group.

Chapter 2
Preventing Denial of Service Attacks on QoS by Pricing of Network Resources

Summary

In this part of the project, we investigated how to use resource *pricing* to discourage or prevent attacks on QoS. The benefits of pricing are flexibility, provable fairness, ease of understanding and implementation, and efficient use of resources. Pricing may also be thought of as a way to discourage wasteful or malicious use of network resources.

We demonstrated how to use pricing for congestion control in TCP/IP and ATM networks. We showed the method can accommodate users with different preferences for stability or maximum quality. We have extended our method to achieve optimum provisioning of resources in DiffServ networks, to meet user needs.

The impact on IETF protocols for call signaling, resource reservation, and resource allocation have been considered, and extensions to one protocol have been proposed. This method has been implemented and tested in an actual network, and source code is available.

Introduction

QoS is provided to applications by making sure they have sufficient resources. The term “resources” refers to many things, including processing time, network bandwidth, memory, storage space, scheduling priority, etc. In our work on QoS, the resource we particularly focus on is network or link bandwidth. The lessons learned apply equally well to other resources.

Control of resources is necessary when there aren’t enough to satisfy all requests. The reasons for this might include:

- many applications are trying to run simultaneously (demand is high)
- the resources are in short supply (e.g.., in combat, wireless bandwidth is limited by battery power and frequency assignment)
- applications scale up to use whatever resources are available (e.g., a file transfer application increases its speed to whatever rate the network will support, up to a maximum speed the file can be transferred to or from the hard drive)
- malicious users deliberately (and spuriously) claim and use resources, just to make them unavailable to other users

Without some sort of control or voluntary cooperation, allocation of resources (and QoS) becomes chaotic or highly unfair. In an environment where attacks may occur, voluntary cooperation cannot be relied upon, and a separate control mechanism is needed.

We believe virtual pricing of resources is an appropriate mechanism for this purpose. By “virtual” we mean that no actual currency has to be used, or exchange hands; any kind of tokens whose supply is controlled may be used. Pricing of a limited resource is easily understood by users, and the principle of price changes in response to changes in the supply and demand of a commodity are well understood also [MMV95]. It has been shown that pricing leads to efficient use of resources. We discuss below the different ways in which pricing is flexible and may support a variety of policies. The method of pricing and bidding for resources is fully distributed and can easily scale to Internet-sized systems. Finally, pricing limits the ability of attackers to acquire and squander scarce resources, and discourages them from attempting to do so.

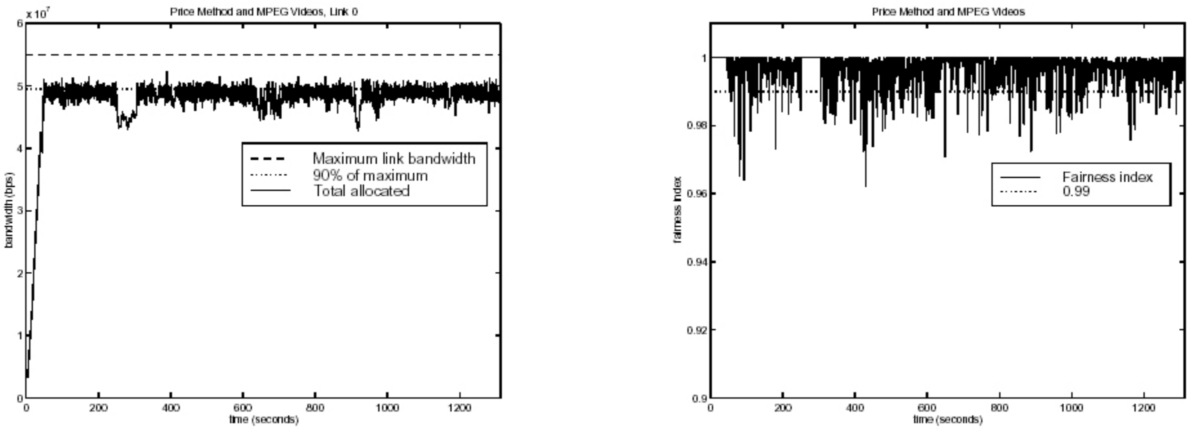
Our method is intended to operate at multiple granularities. At the small end of the scale, it provides a means of allocating a small amount of bandwidth to an application over the span of a few minutes. At the large end of the scale, we discuss how ISPs might use pricing to provision large amounts of bandwidth over weeks or even months.

Methods, Assumptions, Procedures

Allocation of bandwidth for users is done in one of two ways by current Internet protocols. One, the “implicit” approach, is the basis of TCP congestion control. Flows experiencing packet losses are expected to quickly cut back their rate of transmission, followed by slowly increasing their bandwidth until the next congestion point is reached. There is no fixed allocation of bandwidth, so this method is appropriate only for “elastic” applications (those which can tolerate variable bandwidths, such as file transfers). This approach relies on voluntary compliance by the users and enforces a single form of fairness, which is essentially “everybody suffers equally”. The second, the “explicit” approach, requires users to state their bandwidth needs and then reserves that amount of bandwidth using the RSVP protocol. In this method, bandwidth allocation is assumed to be done in a first-come, first-served fashion, and no explicit factors governing fairness are defined.

The essence of QoS is the ability to allocate resources unevenly (more to those that require it, less to those that don’t) based on varying needs of applications, and to enforce that allocation; neither approach above has both of these features. Pricing is a mechanism that has the ability to do so. When an application competes for resources, it must bid for them based on the preferences, demands, and budget of the user running the application. As the demand for bandwidth goes up, or the supply goes down, prices rise and are communicated back to the users. The key aspects of this process are measuring demand, computing prices to achieve close to (but not greater than) 100% utilization of the resources, and distributing prices back to the users. We have shown how to use the protocols of ATM networks, and IP networks, to accomplish these steps. The granularity of bandwidth allocation in this part of the work has been at the level of individual flows or connections, for short timescales (seconds).

We have shown that our method is very practical to implement. Computation of the appropriate price is accomplished by a well-known method referred to as a tatonnement process, which guarantees convergence and optimality under reasonable conditions. The result is efficient management of the resources, and support for a wide range of allocation policies (types of fairness), including max-min, proportional-fair, and utility-fair, and weighted versions of each. It thus has the above-stated desired capabilities (uneven, enforceable allocation based on needs). In the figures below, we show the results of experiments intended to demonstrate fairness and full utilization of bandwidth.



(a) Link 0 allocation graph.

(b) Fairness index graph.

Figure 2. Results of Pricing Experiments showing full utilization and high fairness.

This first step of our approach, however, did not provide guaranteed QoS throughout the duration of an application. In order to accomplish such guarantees, it must be possible to reserve resources at a fixed price. Since this may be regarded as a hedge against future increases in demand (and therefore price), such reservations come at a cost. We have formulated a “multi-market” model in which a mix of users can interact to obtain QoS under varying conditions. This model accommodates both (i) users who prefer the highest possible QoS, at the risk of some variation in resource price and availability, along with (ii) users who will settle for a lower QoS, but who want a guarantee of price stability for a fixed duration.

We further considered the issue of resource allocation for larger-scale aggregations of users, over long timescales. This is the problem of *provisioning* sufficient capacity for the expected demands on the network, considering not just the number of users, but the QoS requirements of those users. We assumed that Differentiated Services (DiffServ) was used to provide long-term, large-scale QoS guarantees, and that ISPs would be responsible for predicting demand, provisioning it (i.e., buying sufficient capacity from a carrier), and then allocating it to individual users at prices determined by our earlier method. We showed how to formulate a hierarchical market model that results in maximum utility for users and maximum profits for the ISP. We then extended these results in two ways. First, we showed that allowing users with lower QoS requirements to “promote” to a higher QoS class when resources are available is a desirable strategy. Secondly, we showed that it is not necessary to adjust prices for users frequently in order to optimize utility. In fact, if price changes are scheduled only 2 or 3 times a day, the loss in aggregate utility (and profit) is generally under 10% under reasonable conditions.

Lastly, we investigated the impact on IP protocols to implement pricing of user bandwidth requests. Both SIP and RSVP are extensible protocols and already have provisions for Authorization Objects that can easily be adapted for our purpose. The primary impact is on the COPS protocol, which has to be modified to support proper interaction between the connection request and resource allocation protocols. A figure of the protocols and their use is shown below.

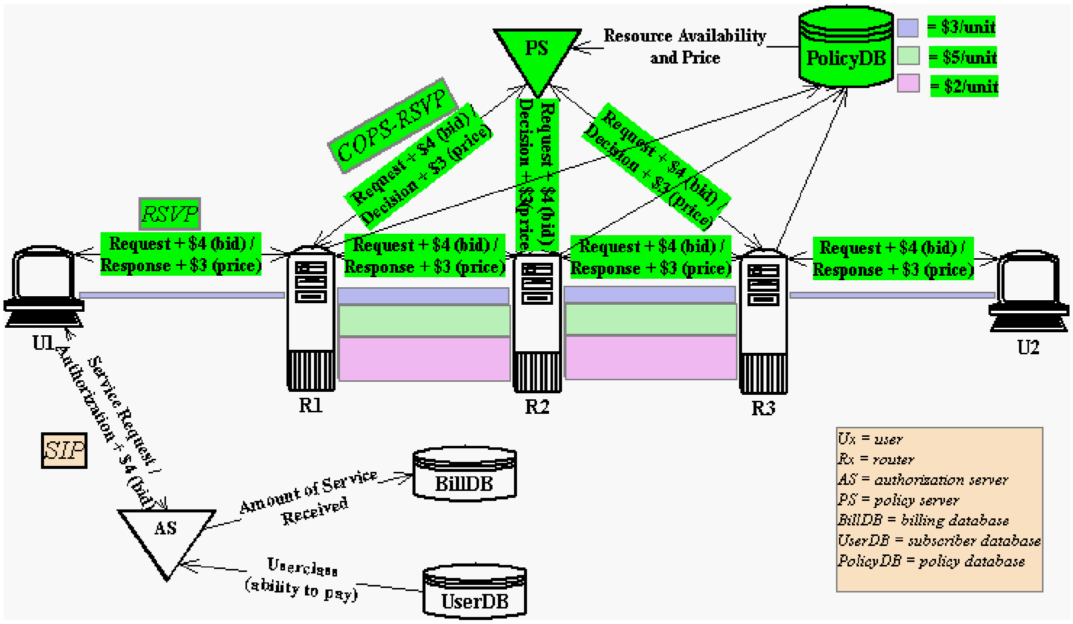

PS
Resource Availability and Price
PolicyDB
🟦 = $3/unit
🟩 = $5/unit
🟪 = $2/unit
COPS-RSVP
Request + $4 (bid) / Decision + $3 (price)
Request + $4 (bid) / Decision + $3 price
Request + $4 (bid) / Decision + $3 (price)
RSVP
Request + $4 (bid) / Response + $3 (price)
Request + $4 (bid) / Response + $3 (price)
Request + $4 (bid) / Response + $3 (price)
Request + $4 (bid) / Response + $3 (price)
U1
U2
R1
R2
R3
SIP
Service Request / Authorization + $4 (bid)
AS
Amount of Service Received
BillDB
Userclass (ability to pay)
UserDB
Ux = user
Rx = router
AS = authorization server
PS = policy server
BillDB = billing database
UserDB = subscriber database
PolicyDB = policy database

Figure 3. Impact of Pricing on RSVP, COPS, and SIP.

Results and Discussion

There have been many outputs of this part of the project. The following presentations have been given, and papers published:

- E. Fulp and D. Reeves, “Distributed Network Flow Control Based on Dynamic Competitive Markets”, presented at and published in the *Proceedings of the International Conf. on Network Protocols (ICNP)* in October 1998.
- E. Fulp, D. Reeves, M. Ott, and D. Reininger, “ABR Rate Control for Multimedia Traffic Using Microeconomics”, presented at and published in the *Proceedings of the International Conf. on ATM (ICATM)* in July 1999.
- D. Reeves, presentation on “Which Goals and What Criteria for Pricing?” at the *Workshop on Pricing and QoS*, ENTS, Paris, in September 1999.
- E. Fulp, invited presentation on “The Cost of QoS: Network Pricing Techniques and Trends” at the NEC Computer and Communications Research Labs, Princeton, NJ in November 1999.
- E. Fulp and D. Reeves, “QoS Rewards and Risks: A Multi-Market Approach to Resource Allocation”, presented at and published in the *Proceedings of Networking 2000*, Paris, France, May 2000.
- E. Fulp, Z. Fu, D. Reeves, and S. F. Wu, “Preventing Denial of Service Attacks on Network Quality of Service”, presented at and published in the *Proceedings of the Defense Information Security Conference and Expo (DISCEX-II),* Anaheim, CA, June 2000.
- E. Fulp and D. Reeves, “Optimal Provisioning and Pricing of Internet Differentiated Services in Hierarchical Markets”, presented at and published in the *Proceedings of the International Conference on Networks (ICN),* Colmar France, in July 2001.

- E. Fulp and D. Reeves, “Optimal Provisioning and Pricing of Differentiated Services Using QoS Class Promotion*”,* presented at and published in the *Proceedings of the Workshop on Internet Charging and QoS Technology (ICQT),* Vienna, Austria, October 2001.
- E. Fulp and D. Reeves, “The Economic Impact of Network Pricing Intervals”, presented at and published in the *Proceedings of the Workshop on Internet Charging and QoS Technology (ICQT),* Zurich, Switzerland, October 2002.
- E. Fulp and D. Reeves, “Bandwidth Provisioning and Pricing for Networks with Multiple Classes of Service”, submitted for publication to the *Computer Networks Journal*, Elsevier Science Publ. In review.

We also integrated our method of pricing in the context of connection establishment with standard Internet protocols (SIP, RSVP, COPS). The outputs are:

- A software release containing modifications of SIP, RSVP, and COPS. This software runs under Linux; both source code and executables are available.
- Louis-Nicolas Hamer of Nortel Networks used some of our ideas about integration of call signaling and resource reservation in his IETF drafts draft-hamer-sip-session-auth-00.txt, draft-hamer-rap-cops-umts-go-00.txt and draft-hamer-rap-session-auth-00.txt.

In September of 1999 Errin Fulp graduated with a PhD in Electrical and Computer Engineering from NC State University. His thesis title was *Resource Allocation and Pricing for QoS Management in Computer Networks*. He is presently employed as an Assistant Professor of Computer Science at Wake Forest University. Khurram Khan described the modifications to COPS for support of pricing in his MS thesis *COPS Usage for Media Authorization*, for which he received the MS in Computer Science from NC State in November, 2002. He is currently employed by Spirent Networks.

Conclusions, Recommendations

We believe pricing is an appropriate method of allocating resources and discouraging or preventing attacks on QoS. Our work has shown analytically and experimentally that this is so. We have also addressed the issues of protocol implementation and signaling in support of our pricing methods. We have not, however, investigated methods of billing and accounting in support of pricing, which are topics that await solutions and standardization by IETF.

References

[MMV95] J. K. MacKie-Mason and H. R. Varian. “Pricing congestible network resources”. *IEEE Journal on Selected Areas in Communications*, 13(7):1141 -- 1149, Sept 1995.

Chapter 3

Protection of Reliable Multicast Against Receiver-Based Attacks

Summary

Reliable multicasting is used to send a stream of data to a set of receivers efficiently, at the same time, and at the same rate. With reliable multicasting, it is possible for a malicious receiver to intentionally slow down or even effectively halt transmissions to all receivers in the multicast group.

In this part of the ARQoS project, we investigated the problem experimentally and showed that this vulnerability is severe and easy to exploit. We proposed a theoretical solution to the problem by using auctions (specifically, the Generalized Vickrey Auction) to discourage malicious behavior by receivers. We then investigated the practical issues of supporting such an auction-based solution in a standard reliable multicast protocol, PGM. We modified PGM and simulated its behavior with these modifications.

Introduction

Multicasting is used to deliver data to a set of receivers connected to the Internet in a way that is more efficient than sending the data separately to each receiver. *Reliable* multicasting [LG98] is required when all receivers must receive exactly the same data, as is the case for many types of critical data. Examples of military applications that would benefit from reliable multicast are distributed real-time simulations (or wargaming), and hierarchical control systems.

The figure below shows an example of a multicasting *tree* with one sender and 4 receivers. Those routers (green circles) which receive one copy of the data and forward multiple copies to the next level of the tree are called *multicast routers*.

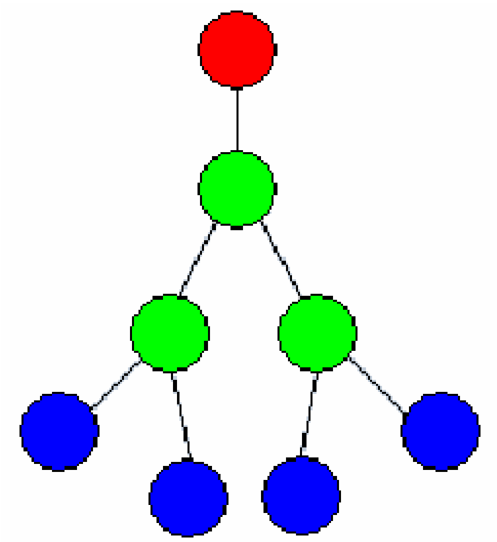


Figure 4. A Multicast tree.

If all receivers get the same data, and the data is sent once by the source, and the multicast routers can buffer only small amounts of data, reliable multicasting can only be accomplished if the receivers receive the data at roughly the same rate. If the rate of transmission is determined by feedback from the receiver, as with TCP, this means that reliable multicast runs at the speed of the *slowest* receiver. Unfortunately, this creates the opportunity for malicious users to slow down or even halt reliable multicast transmissions for all receivers in a group. The malicious user need simply lie when reporting it can't keep up with a higher transmission rate.

To investigate the severity of this problem, we simulated reliable multicasting for a specific, standard protocol, PGM (Pragmatic Generic Multicast) [RFC3208]. In PGM, receivers send NAKs (negative acknowledgments) to the sender when they notice they missed some packets. Any multicast router receiving a NAK can send the missing data if it has it; in the worst case, a NAK propagates back to the source, which is responsible for resending the missing data. There are two approaches to flow control in PGM, one of which is paced by the NAKs received from the receivers. We simulated the behavior of PGM for different NAK generation rates by the receivers, for a small multicast tree with 15 receivers. The results are shown in the figure below, and demonstrate the sensitivity of reliable multicast to the rate at which receivers generate NAKs. A multicast tree with hundreds or thousands of receivers would exhibit much worse behavior than this example, which illustrates the vulnerability of reliable multicast service.

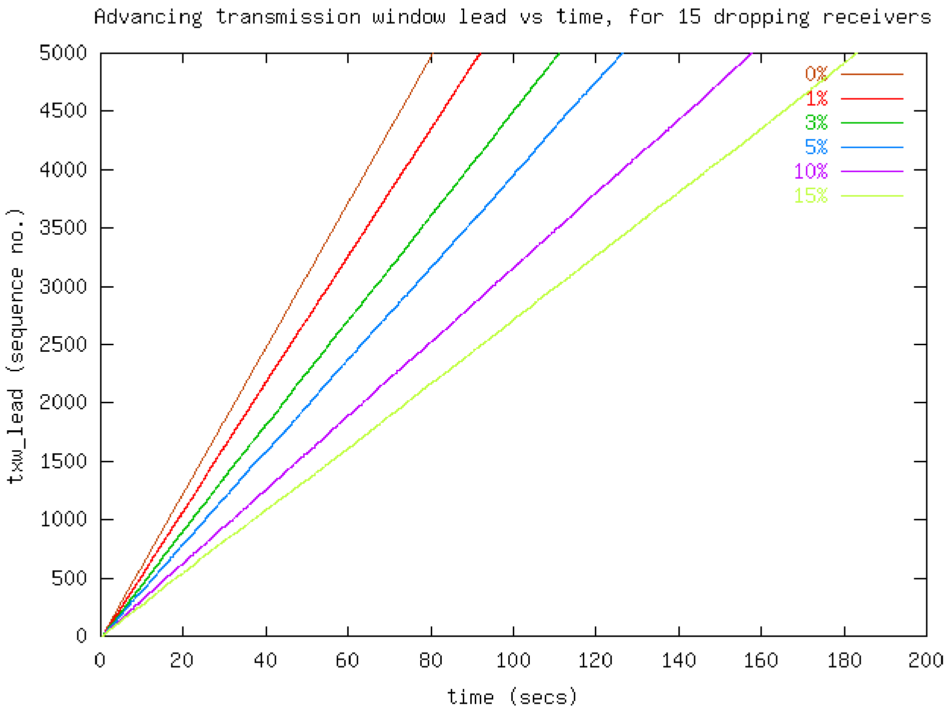


Figure 5. Effect on transmission time of receiver packet loss rates.

Methods, Assumptions, Procedures

As in our pricing work, our strategy is to discourage attacks on reliable multicast, by giving users (receivers) incentives to state truthfully what their receive capabilities are. Such incentives don't absolutely prevent falsifying feedback (NAKs) to the sender, but make it difficult or *expensive* for the attacker to do so. We assume that the receivers have to state their preferences for transmission rates, and submit *bids* with those preferences. These preferences and bids are collected by the sender, and a *Generalized Vickrey Auction (GVA)* [VMM95] computation is then performed to determine the optimum transmission rate. GVA maximizes the aggregate utility of all receivers, is straightforward to compute, and is incentive-compatible, which means that the optimum strategy for all receivers (including malicious receivers) is to state truthfully what their preferences are. In essence, a receiver desiring a lower transmission rate must compensate for the loss in value to other receivers who prefer a higher transmission rate.

This result is an elegant solution to the problem of attacks on reliable multicast, but a major challenge remains: tailoring it to work well with an existing reliable multicast protocol. We modified two of the existing PGM messages to include bid information from the receivers to the sender, and the resulting rate / price information from the sender back to the receivers. We also implemented the auction computation at the sender, and policing of the receivers to verify that they conformed to the expected receive rate. We simulated this method and showed that the attack on reliable multicast is defeated by our modifications; users who request excessive retransmissions (i.e., who claim not to receive packets at the negotiated rate) will not cause a slowdown in this negotiated rate.

Results and Discussion

We believe that work on reliable multicast has failed to address this issue of receivers who deliberately slow down the transmission rate in order to impact other receivers. We have presented an elegant solution based on auction theory, and we have shown how that solution can be integrated with an existing reliable multicast protocol. Our contributions are:

- Identifying the likelihood and impact of the attack
- Presenting a theoretical solution
- Showing a practical implementation of that solution
- Demonstrating the effectiveness of the implementation

Two faculty (Douglas Reeves, Peter Wurman) were involved in this work. Some outputs of the work are:

- A paper was presented at the International Workshop on Internet Charging and QoS Technologies in Zurich, Switzerland in October 2002, and published as A. Sureka and P. Wurman, "Applying Generalized Vickrey Auction (GVA) for Pricing Reliable Multicasts", in Proc. Of the 2nd International Workshop on Internet Charging and QoS Technologies (ICQT 2002), October 2002, pp. 282—292.
- The experiments on PGM, and modifications to PGM, were done as part of the ns-2 network simulator. The code is on our website.

Two students received their MS degrees in Computer Science. Ashish Sureka graduated in MS in Computer Science from NC State in May 2002, and is continuing on for the Ph.D. under Dr. Wurman's direction. Nipul Shah defended his thesis entitled *Preventing Denial of Service Attacks on Reliable Multicast Networks* in the Department of Electrical and Computer Engineering at NC State, and is currently employed by Spirent Networks in the development and testing of advanced networking products.

Conclusions and Recommendations

Reliable multicast has been proposed in a number of papers, and has been standardized as multiple protocols. To date, adoption of both unreliable and reliable multicast has been slow. It has been suggested that reliable multicast should be performed at the application layer. In this model, the function of the multicast routers is performed by application gateways instead, and there is no need to deploy multicast support in the network. However, even if application gateways are used for reliable multicasting, there is still a high probability that malicious users will attempt to disrupt or degrade transmission speed by lying about the rate at which they can receive data. Our insights and methods will be just as important in this case, and even more feasible to implement, since providing the proposed functionality at the application layer imposes fewer constraints on backwards compatibility.

If existing reliable multicast protocol standards are to be robust and resistant to attack, they will need to be modified. We present our approach as a specification of how this can be done.

References

[LG98] B. Levine and J. Garcia-Luna-Aceves, “A Comparison of Reliable Multicast Protocols”, *ACM Multimedia Systems*, 6, pp. 334-348, 1998*.*

[RFC3208] T. Speakman et al., “PGM Reliable Transport Protocol Specification”, IETF RFC 3208, Dec. 2001.

[VMM95] H. Varian and J. Mackie-Mason, *Generalized Vickrey Auctions*, Technical report, University of Michigan, 1995.

Chapter 4

Capacity Planning for Networks Supporting DiffServ and MPLS

Summary

Differentiated Services (DiffServ) and Multiprotocol Label Switching (MPLS) are two popular protocols for implementing QoS in the Internet. Both depend on careful design of the network to provide adequate capacity for expected user traffic demands. Existing methods of network design, or capacity planning, do not consider quality-of-service constraints for multiple classes of service. We have proposed a method of capacity planning that extends existing optimization methods to the case of multiple class of service traffic demands. The method is designed specifically for DiffServ and MPLS-enabled networks. Our evaluation has shown the practicality and usefulness of this technique. Recent work has demonstrated how the method can be modified to ensure sufficient capacity is also available to tolerate network failures while providing QoS.

Introduction

There have been many proposals for adding quality of service (QoS) guarantees to the Internet, but most have met with resistance. The two most popular proposals at present are to provide only a few classes of service (via Differentiated Services, or DiffServ) and to allow more fine-grained control of routing and capacity allocation (via *Multiprotocol Label Switching*, or *MPLS* [RFC3031]. In order to result in proper control of QoS, both require careful attention to designing the network to have sufficient capacity for the predicted usage. This is because DiffServ only provides relative and not absolute QoS, and does not implement per-flow admission control. Without proper network design, the amount of traffic entering the network may overwhelm the capability of providing differentiated services.

The methods for doing so are referred to as “capacity planning”, for which well-established methods of optimization exist. The three aspects of capacity planning are:

- Topology design – determining location of routers and what routers are connected by direct links
- Link sizing – determining how much bandwidth is needed for each link in the network
- Routing – determining what path traffic takes through the network to reach its destination

In most cases, topology design is determined by business factors, and the resulting topology is input to an optimization method that solves the link sizing and routing problems. Such methods to date have treated all traffic as having the same end-to-end delay and loss requirements, which is not appropriate if some applications require better QoS than others.

Methods, Assumptions, Procedures

In our initial work, we formulated a new version of the capacity planning problem for networks using Differentiated Services. We assumed that some fraction of the traffic was in the highest quality (EF) class, and the remaining traffic was in the conventional (BE) class. In our model, EF traffic must receive the highest priority service, but without penalizing the performance of BE traffic beyond acceptable limits specified by the user. This results in a new constraint on the optimization process.

The resulting problem may be formulated as a non-linear integer programming optimization problem, which is normally very expensive to solve for large problem sizes (keep in mind that capacity planning may need to be done for networks with hundreds to thousands of routers and links). The non-linear aspect is due to the QoS constraints, while the integer programming aspect is due to the restricted set of choices of link sizes. One of our major innovations in this work was suggesting how the problem could be decomposed, or separated, into smaller problems that could be solved individually, and whose solutions could be combined to provide a global solution. The resulting blend of Lagrangean optimization with gradient descent and use of the subgradient method is a sophisticated and elegant solution method.

Our initial work did not depend on or exploit the capabilities of MPLS, which allows classifying traffic and assigning it to specific paths through the network. It is widely expected that network administrators will make use of MPLS to more carefully control and engineer their networks. In the second step of our work, we added to the initial problem formulation the additional capabilities of MPLS. This significantly complicated the problem, but we have shown that our solution technique can be modified to incorporate this capability. The resulting work has just been presented at the International Teletraffic Congress, one of the premier conferences on capacity planning and performance analysis for QoS.

Most recently, we have looked at the issue of supporting QoS in capacity planning, while at the same time providing adequate spare capacity to tolerate failures in the network. Such failures occur in normal operation, and must be planned for. They also may occur due to abnormal causes (e.g., the September 11 attack took out a major chunk of Internet capacity in the northeastern US), and may be expected in networks operating under military control during times of conflict. We are working on adding to our optimization method the requirements to maintain connectivity, and QoS, despite a limited set of network failures. Our recent results are promising, and almost ready for submission.

Results and Discussion

We applied our method to the problem of optimizing link capacity and routing traffic in such a way that QoS constraints of EF and BE traffic are both met. The experimental inputs were a variety of representative network topologies, with as many as 1000 nodes and 2500 links, and user traffic demands, up to 40,000 in number. The solution quality was within a few percent of the optimal solution, which is outstanding. The running times are shown below, which demonstrate our method is practical to use for large networks. It is the first successful approach to capacity-planning for networks supporting traffic with multiple QoS requirements.

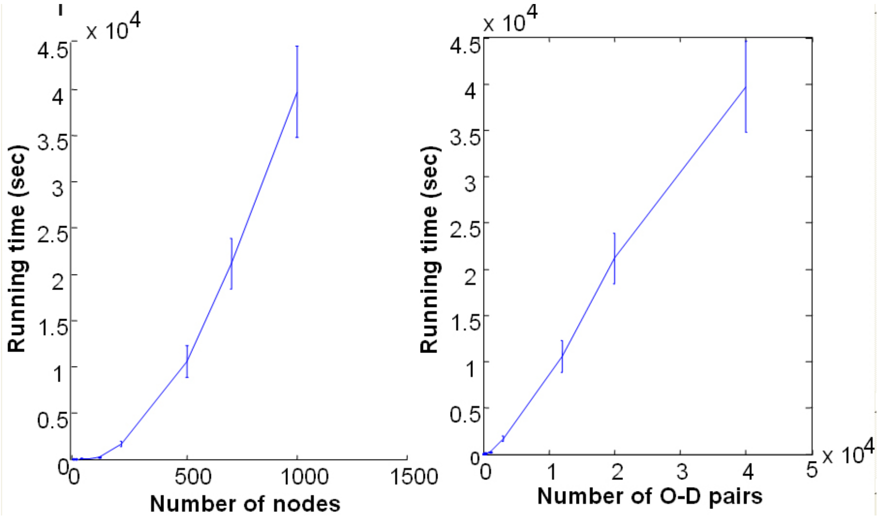


Figure 6. Running time of the DiffServ capacity planning heuristic.

As mentioned, similarly positive results have been reported for the case of networks supporting both MPLS and DiffServ, and we are currently collecting results on networks able to continue providing QoS despite failure of some links and/or routers.

The outputs of this work are the following:

- A presentation at the International Conference on Communications (ICC) in Anchorage, AK in April 2003, published as: K. Wu and D. S. Reeves, “Capacity Planning of DiffServ Networks with Best-effort and Expedited Forwarding Traffic”, in *Proc. Of the International Conference on Communications,* May 2003.
- A presentation at the International Teletraffic Conference (ITC-18) in Berlin, Germany, published as: K. Wu and D. S. Reeves, “Link Dimensioning and LSP Optimization for MPLS Networks Supporting DiffServ”, in *Proc. Of ITC-18 (International Teletraffic Conference),* September 2003*.*
- K. Wu and D. S. Reeves, “Capacity Planning of DiffServ Networks with Best-Effort and Expedited Forwarding Traffic”, to be published in *Telecommunication Systems*, January 2004.

Kehang Wu will graduate with a PhD in Computer Engineering from NC State University in December 2003. The subject of his thesis is the work described in this chapter. His plans after graduation have not been settled at this time.

This work has benefited from collaboration with Dr. Cole Smith of the University of Arizona, another DARPA FTN PI, and by Dr. Michal Pioro of the Warsaw (Poland) University of Technology.

Conclusions and Recommendations

Methods of optimization for capacity planning have been well developed and widely applied for networks supporting just one class of traffic (like voice traffic, or data (best effort) traffic). We have shown how to improve them for capacity planning of networks supporting multiple classes of traffic, under multiple constraints (use of MPLS, adequate spare capacity). The results should be useful for providing QoS at low cost, while requiring only modest new capabilities from the network.

References

[RFC3031] E. Rosen, A. Viswanathan, and R. Callon, “Multiprotocol label switching architecture”, IETF RFC 3031, January 2001.

Chapter 5
Detection of Denial-of-QoS Attacks on DiffServ Networks

Summary

In this part of the ARQoS project, we proposed a method of detecting attacks on Quality of Service in DiffServ networks. Our approach aims to achieve real time operation, fast detection, and scalability to large networks. Sensors sample QoS metrics like bit rate, packet dropping rate, and jitter of Virtual Leased Line (VLL) flows at strategic points in their paths. We detect anomalies in sampled network flow statistics using the EWMA Control Chart test for metrics that are highly stationary, and for the rest we use a method based on SRI's χ^2 test (NIDES). We also use rule-based intrusion detection of SLAs as a complement to these techniques.

We have tested our intrusion detection methods using emulation on a testbed, and using simulation. Attacks are detected 100% of the time, and require from under a minute to approximately 15 minutes to detect. The false alarm rate at the sensitivity level used to achieve these results was less than 1%.

Introduction

Differentiated Services, or DiffServ, is a proposed standard for providing QoS in a lightweight way, to aggregations of flows. The benefit of DiffServ is that it provides generally “good enough” QoS, at substantially less cost and complexity that Integrated or Guaranteed Services. On the other hand, because DiffServ does not control QoS as tightly as IntServ, it is more difficult to determine if the network is providing the desired QoS.

Routers at the boundary, or exterior access point, of a large network will classify, mark, and police incoming DiffServ traffic. Routers in the interior of the network are responsible for providing the requested QoS. Two customers may be connected by a *Virtual Leased Line* (*VLL*) through a DiffServ network. The VLL pins down the route taken through the network, helping to ensure that quality will not vary during the lifetime of a connection.

Following are some of the DiffServ-specific attacks which we wish to detect and defend against:

- A malicious external host can attempt to flood a boundary router, in effect denying service to other users of the DiffServ network.
- A compromised DiffServ interior (core) router may remark, drop, or delay packets in violation of their QoS contract.
- A compromised interior router may also flood the DiffServ network with bogus traffic.

The challenge in this part of the project is to distinguish the normal, small variability in QoS (delay and loss rates) from the perturbations caused by deliberate manipulation of and interference with the functions of DiffServ routers.

Methods, Assumptions, Procedures

Detecting attacks on QoS requires measurement of QoS. We have implemented the following "sensor" programs for this purpose. First, sensors record the packet and bit rates of aggregated traffic and individual VLLS every 10 seconds, at each router. Secondly, to have a "controlled" source to monitor, we inject probe traffic at a specific, low rate into an existing VLL. The probe packets are cryptographically signed to be recognizable and unforgeable. Sensors then monitor the packet rate and delay variation (jitter) of this probe traffic. A system diagram is shown below.

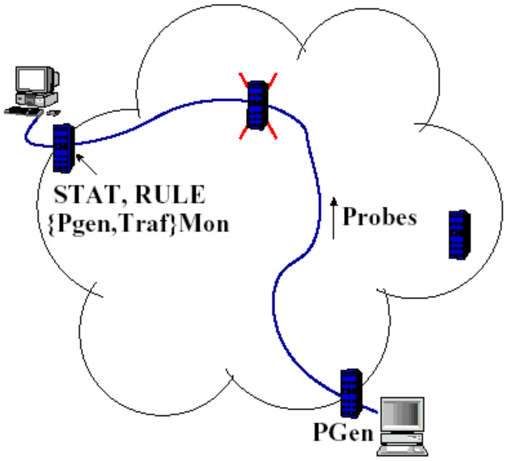


Figure 7. Placement of sensors and probes for DIffServ Intrusion Detection

Statistical anomaly detection is based on the hypothesis that a system will behave anomalously when attacked and that these anomalies can be detected [D87]. A flow that has contracted for a specific QoS is a good candidate for the use of anomaly detection methods. The anomaly detection approach is preferred for detection of new or uninvestigated attacks with undefined attack signatures -- as is the case with attacks on DiffServ.

The statistical anomaly detection algorithm we use is based on SRI's NIDES algorithm [JV93], which tries to fit QoS measurements to an expected distribution using the χ^2 test. The behavior of DiffServ flows not under attack are measured over extended periods of time to capture the normal (non-attack) variation in the DiffServ network. The distribution is discretized into "bins", each representing a range of values for delay, delay variation, packet loss rate, throughput, etc. After every preset long term update period, QoSSTAT updates the long term distribution using exponentially weighted moving averages. The short term profile is updated every time QoSSTAT receives a new report of a measured value. In our tests we use a short term estimation period of 15 minutes (900 seconds), with one new measurement every 10 seconds.

Our method maintains a long term χ^2 distribution curve, against which each new short term χ^2 value is compared to calculate the anomaly score *S*. A user-specified threshold indicates what probability level will be classified as "anomalous". A higher threshold will favor quick and thorough detection of attacks, but will also mean an increase in the number of *false alarms* (*false positives*). In our experiments, we specified two thresholds:

- Anything with probability below .1% was interpreted as a definite attack
- Anything with probability between .1% and 1% was interpreted as a possible attack.

Certain parameters such as the drop rate associated with an EF probe flow in an uncongested network may be highly stationary. The χ^2 test in general will not work well for such parameters. For detecting anomalies in highly stationary performance measures, we propose the use of an alternative method based on Control Charts [R59]. In statistical process control, the EWMA control chart is used to detect small shifts in the mean of a process to indicate an out-of-control condition. In contrast, we monitor flow processes that are only weakly stationary, and hence have to allow for a gradual shift in their means over sufficiently long periods of time. However, a swift shift will be detected as an anomaly and flagged. The long-term average is computed using an exponentially-weighted moving average, and compared with a short term estimate. If the short-term estimate is more than 3 standard deviations away from the long-term average, it is regarded as evidence of an attack.

The statistical anomaly detection approach has been criticized for having a higher false alarm (false positive) rate than the rule or signature-based detection approaches. In view of this, we also use a simple *rule-based* detection technique, henceforth referred to as RULE. It is designed to quickly detect obvious violations of the SLA, based on DiffServ specifications [RFC2475]. The attack signatures we use include the packet dropping rate exceeding a specified rate for the Assured Forwarding (AF) probing traffic, EF packets being remarked instead of dropped when they are delayed, etc.

We thus have 3 methods of detecting attack on QoS in DiffServ networks: the χ^2 test, EWMA, and RULE.

Results and Discussion

We ran experiments to investigate the effectiveness of our intrusion detection method at detecting attacks. These experiments were done in two ways. In the “emulation" tests, we used PCs running Linux as hosts and routers in a dedicated (and small) testbed. In the “simulation" tests, we used the simulator ns-2 to take additional measurements and investigate more complex traffic models.

We created a variety of types of attacks to test our intrusion detection methods. The attacks were of either the persistent or intermittent types. Persistent attacks were continuous for a specific duration, while intermittent attacks last awhile, stop for a period, and then repeat the cycle.

We set up a VLL between an ingress and an egress router, carrying video traffic. In one test the VLL received EF service, and AF in another test. We generated "background" traffic using standard traffic generators. We intentionally congested the routers to induce some "normal" variation in end-to-end delays. We implemented the following attacks on the VLL traffic:

- increase end-to-end delay and delay jitter
- increase packet dropping (from .5% to 5%)
- remark to a lower QoS level 10% of the traffic

All attacks in the testbed were of the persistent type, and were successfully detected, with detection times ranging from 50 to 800 seconds.

In the simulation experiments, the background traffic generated was self-similar, to represent a more challenging test case. Both persistent and intermittent attacks were mounted. All attacks were detected, with detection times ranging from 80 to 900 seconds.

Our anomaly detection methods have non-zero false positive rates arising from the thresholds used for alarm detection, and the variability in normal (non-attack) traffic. The χ^2 tail probability α and the 3 σ limits were chosen in our work to produce a 1% chance of a false positive for a monitored flow that is Normally distributed.

The software developed as part of this project is available on our website for download and testing. This software implements the 3 detection methods, the sensors, and the probe packet generation. It is designed to run on the Linux operating system.

Vinay Mahadik graduated with a MS in Computer Engineering in August 2002. The title of his thesis was *Detection of Denial of QoS Attacks on DiffServ Networks*. He is currently employed at Qualys, Inc.

Conclusions, Recommendations

Our intrusion detection system detects QoS degradation quickly and in real time. All the identified attacks were detected. The possibility of generating a false alarm is low and can be tolerated in most deployments. The sensors and the detection engine are all light weight threads that can monitor several VLLs at a time. Our work is the first reported on intrusion detection for QoS attacks. We advocate the use of methods such as this as QoS mechanisms are deployed on a widespread scale.

We believe these results make our work a strong candidate for deployment, and that the vulnerability of DiffServ to attack justifies attention to the issue. This is the first work on intrusion detection for security of QoS.

References

[D87] Dorothy E. Denning. “An intrusion detection model”, *IEEE Transactions on Software Engineering*, SE-13(2):222--232, Feb 1987.

[JV93] H. Javits and A. Valdes. “The NIDES statistical component: Description and justification”, Technical report, SRI International, Computer Science Laboratory, March 1993.

[R59] S. W. Roberts. “Control chart tests based on geometric moving averages”, *Techometrics*, 1(3):239--250, 1959.

[RFC2475] S. Blake et al., “An Architecture for Differentiated Services”, IETF RFC2475, December 1998.

Chapter 6

Statistical Anomaly Detection for TCP Packet Dropping Attacks

Summary

TCP is used as the major transport service in the Internet. Congestion in the Internet requires users and routers to cooperate responsibly to solve the problem. Because there is no central, verifiable enforcement of this responsibility, malicious users and routers can negatively impact the TCP throughput of other users, in an unnecessary and undesirable way. The challenge in detecting such attacks is that *some* packet dropping is normal, and necessary in dealing with congestion.

In this part of the project, we implemented a method of detecting TCP packet dropping attacks using statistical analysis of the packets dropped, as measured by the host. The method is scalable and fairly lightweight. We implemented a variety of packet dropping attacks and tested our method under realistic conditions. The results indicate our method is effective at detecting significant levels of malicious packet dropping, with fairly low false positive rates.

Introduction

TCP provides a reliable transport service on top of IP networks, and is the dominant transport service in the Internet. When congestion occurs, routers inevitably must drop, or discard packets due to insufficient buffer space. TCP also copes with congestion in the Internet by requiring end systems or hosts sharing a bottleneck link to throttle back their sending rates to equal, sustainable levels.

The assumptions by users are that routers treat all users fairly (equally) during times of congestion, and that all users scale back their traffic rates equally during times of congestion. Either of these assumptions may be violated, either by malicious routers, or by malicious users. The result is an undesirably, and unnecessarily, high level of packet discarding for non-malicious users. Routers, in particular, which have complete and unrestricted control of packet forwarding and dropping, can easily implement attacks against user's traffic. Detecting attacks against the throughput of TCP flows has not been addressed by previous work on intrusion detection. In this part of the ARQoS project, we investigated and implemented a method of detecting such attacks. Criteria for evaluating such a method include:

- The method should be scalable. With increasing link speeds, routers switch traffic from an ever-increasing number of flows. A method which uses the hosts attached to the Internet as data collection and analysis points will scale better than a method which requires traffic monitoring and analysis by routers.
- It should be effective at detecting artificial (non-congestion-related) packet dropping, and be unaffected by the normal congestion-related packet dropping that occurs in the Internet.
- The method should have modest computational requirements.

We have worked on a method for detecting packet-dropping attacks using the techniques of statistical anomaly detection. We describe the method below, followed by our experimental results.

Methods, Assumptions, Procedures

We began by investigating how effective TCP packet dropping attacks of various types might be. For experimental purposes we transferred the same file from four different sites, two in Asia, one in Europe, and one in the U.S. Packet dropping was implemented by a machine acting as a router in our lab. All other aspects of transmission through the Internet were unaffected. The packet dropping attacks were of the following types:

- Periodic – every nth packet was dropped.
- Retransmissions – a specified number of times a particular packet and all retransmissions of that packet are dropped.
- Random – the packets dropped are uniformly randomly selected.

The Retransmission attack was the most disruptive to TCP throughput and delay, for the same number of packets being dropped. The throughput of the TCP session could be decreased several-fold with just a few packets picked for retransmission attacks.

Our method of intrusion detection was based on SRI's NIDES/STAT algorithm. This algorithm continuously monitors a system's behavior, and generates an alert or alarm when the system's current (short-term) behavior deviates significantly from its expected behavior. The expected behavior is based on construction of a long-term *profile* collected over a period of time when the system is not under attack. The profile captures some of the normal variability in the behavior of the system.

The number of TCP packets dropped over a long time period is measured during the training phase, and the PDF of these packet counts is constructed. Then for each successive short-term time interval, a measure of how closely the short-term PDF agrees with the long-term PDF is calculated. Under certain assumptions, the variance between these two should have an approximate χ^2 distribution with n-1 degrees of freedom. In NIDES/STAT, the long-term profile training consists of three phases: category training (learning the subject's expected behavior), Q training (learning the empirical distribution of the variance between the long-term and short-term behaviors) and threshold training (establishing an appropriate threshold for detecting anomalous behavior).

We implemented a method of detecting packet dropping attacks which was based on the statistical analysis methods of SRI's NIDES/STAT. We call our method *TDSAM*, for TCP dropping statistical analysis module. In our method, three metrics — total ftp connection delay, the position of packet reorderings, and the number of packet reorderings per connection — were used as input to the statistical analysis model for intrusion detection. We hypothesized that the connection delay, the position and number of packet reordering would all deviate significantly from their corresponding normal patterns under a dropping attack. The long-term profile was constructed by establishing 20,000 consecutive TCP connections to each of the 4 sites. The short-term profiles were collected from 5,000 consecutive FTP connections. The Q values were distinctive for each site; the figure below shows the Q distribution for the European and the US sites, for instance.

After establishing the long term profile and measuring the *Q* distribution, we subjected the ftp connections to attack using our attack agent. An important value called the *red threshold* was defined for the *Q* distribution, which is the frequency a given short-term measurement occurred during the training phase. For the experiments, we set this threshold to 1%.

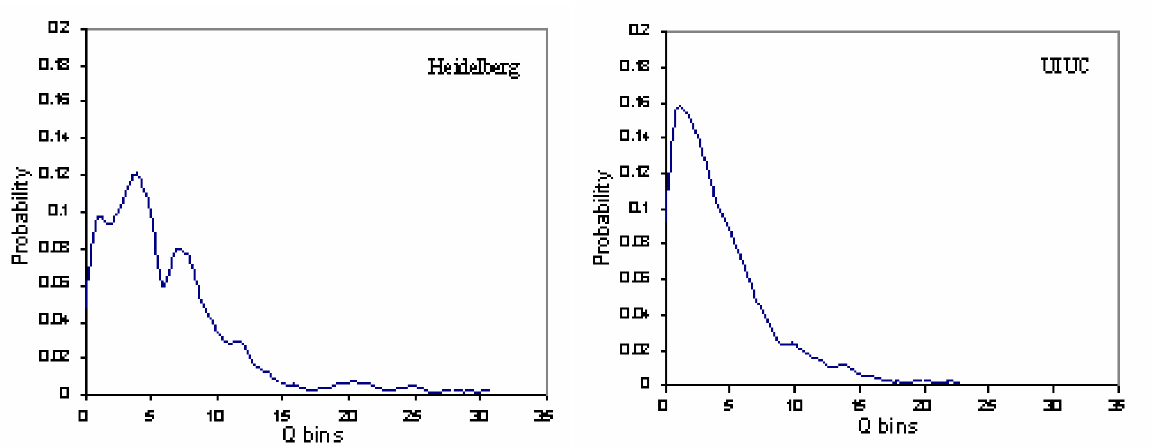


Figure 8. Q Distribution of packet dropping rates for two FTP sites.

Results and Discussion

The results of our experiments showed that TDSAM has a high detection rate for most Periodic dropping attacks. This is because normal packet reordering nearly uniformly distributes across a connection, but periodic dropping attacks generate a very different packet reordering distribution. The Random packet dropping attack is not as easy to detect, however, as it more nearly matches normal (congestion-based) packet dropping. Retransmission dropping attacks are the easiest to detect. The results show that attacks causing the most harm to TCP connections are those which are easiest to detect with our method, which is the desired-for result. For all 3 types of attacks, if all 3 metrics (delay, position of reorderings, count of reorderings) were combined, the false positive (false alarm) rate was generally under 1%.

We conclude by noting that we are measuring performance only at the receiver side; no cooperation is required by the sender, or from the network. For this reason, we believe our method is quite general. Our results demonstrate that detection of dropping attacks by host systems attached to the network can be very successful.

The outputs from this part of the ARQoS project include:

- A paper entitled “Effect of Malicious Packet Dropping on TCP Performance, and Detection of Such Attacks”, presented at and published in the *Proceedings of the International Conf. on Network Protocols (ICNP)* in November 2000. The paper was authored by Xiao-Bing Zhang, S. Felix Wu, Zhi Fu, and Tsung-Li Wu.
- We presented this work at the DARPA Information Survivability Conference and Expo (DISCEX II) in 2001 in Anaheim, CA. The paper was published as: E. Fulp, Z. Fu, D. Reeves, and F. Wu, “Preventing Denial of Service Attacks on Quality of Service”, *Proc. of DARPA Information Survivability Conference and Exposition (DISCEX II),* June 2001.
- The software developed in this project has been packaged and released. It is written in C for the FreeBSD or Linux platforms. Both source code and executables are provided. The code includes packet recording software, packet dropping (attack) software, and the intrusion detection software, including the long-term profile training component.

Xiaobing Zhang defended his MS in Computer Science thesis on *TCP Packet Dropping Attacks and Intrusion Detection* in May 2000. He is currently employed by Ericsson in Raleigh, NC as a staff engineer.

Conclusions and Recommendations

The security and intelligence communities expect that an increasing number of attacks will be on the network infrastructure, and that we must prepare for such attacks. If, in fact, attackers can gain control of routers, then one of the ways they could disrupt or degrade network operation is to drop TCP packets. It is likely that they will do so in such a way that the mechanism or source of the attacks is non-obvious, but the effects of the attack will be severe. We have shown that TCP packet dropping attacks are easy to mount, and that with small manipulations of the packet stream, user throughput can be severely degraded. We have also shown that detection of such attacks is possible by means of statistical anomaly detection methods.

We recommend that hosts be recruited to monitor the network performance and alert network administrators when the network performs in ways outside the normal, expected bounds. By this means a variety of attacks, including packet dropping attacks, may be detected and isolated in the minimum amount of time.

Chapter 7

Automatic Generation Of IPSec/VPN Security Policies In An Intra-domain Environment

Summary

Security of communication in the Internet is being largely handled by IPSec [RFC2401]. IPSec may be used to set up secure tunnels between two endpoints in the network. These tunnels provide encryption of traffic, authentication of traffic, or both. Configuring tunnels in a large enterprise to meet the needs of the organizations is tedious, difficult to understand, and error-prone.

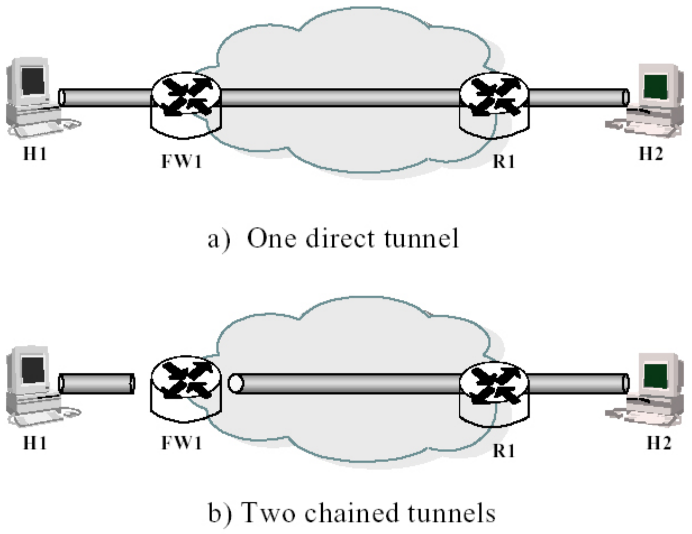
In this work, we allow the security administrators to specify the high-level security requirements for their networks. For this purpose, we have designed a security policy specification language, which is both sound and complete. From this language, we can then generate automatically, or synthesize, a set of IPSec tunnels and policies for use of those tunnels that will satisfy the specified high-level security requirements. We demonstrate that the generated tunnels are both correct and efficient, and that the synthesis process runs in reasonable time for large network configurations. The method saves user time, and is much less likely to have security gaps or vulnerabilities than manual configuration.

Introduction

IPSec [RFC2401] is the preferred method for ensuring secure communication in the Internet. Because it is implemented at the IP layer, it provides this service to all applications that require the traffic to be encrypted or that require assurances of message integrity. Before IPSec is used, a set of *Security Associations* (*SAs*) must be established between points (hosts, security gateways) in the network. Each SA specifies the type of security to be implemented, the algorithm to use, and to what traffic the security should be applied. (In the following, we refer to a SA also as a security *tunnel* between two endpoints, or alternatively, as a security *policy*, following common usage in the security community.)

It is not difficult, given current administration tools, for the user to manually specify the set of SAs that he or she wants and have them implemented correctly. What is difficult, however, is to ensure that the set of SAs correctly implements a high-level security requirement that the user's organization wishes to enforce.

At first glance, it might seem that a security requirement would map directly and easily to an IPSec SA or tunnel. The following example, however, illustrates the difference between them. In the figure below, two hosts are connected securely in two different ways (by one tunnel in figure A, and by two tunnels in figure B). Suppose an additional security requirement for this network is that traffic from host 1 to host 2 must be visible to (unencrypted at) the firewall, so that it can be examined for viruses and other undesired content. In this case, only the configuration in figure B satisfies both requirements. A user who is unaware of this additional requirement will not know that the SA he or she sets up is inconsistent with this additional requirement, and prevents it from being satisfied. Also, a user who does not know the network topology (existence of the firewall on the path from host 1 to host 2) will not know how to decompose the tunnel in figure A into the two tunnels of figure B. There are many more complex examples that could also be used to illustrate how users setting up tunnels individually may inadvertently violate each other's security requirements, and how knowledge of the network topology is necessary to resolve such violations.



a) One direct tunnel

b) Two chained tunnels

Figure 9. Example illustrating problems in IPSec tunnel configuration.

Detecting and resolving conflicts, and configuring SAs to meet a set of security requirements, is difficult, tedious, and error-prone when done manually. In our work for this project, we have tackled multiple aspects of this problem:

- How can users specify their security needs in a language that is complete, but simple to use?
- How can conflicting security requirements (i.e., that cannot simultaneously be satisfied by any configuration) be detected and resolved?
- How can a specific set of SAs or tunnels be proved to either satisfy or fulfill a high-level security specification, or to violate it?
- Given a high-level security specification, how can a minimal set of tunnels or SAs be automatically configured or generated, such that the security specification will be satisfied by that configuration?

The methods we propose, and their benefits, are described below.

Methods, Assumptions, Procedures

The starting point of this project was to create a method for users to specify what their high-level security requirements are. We designed a language that we believe captures the most important security requirements in a compact, but understandable, way. Traffic flows are specified by means of patterns, or filters, that identify protocol types, ports, etc., and network elements (hosts, routers, firewalls, gateways) are specified by specific addresses or ranges of addresses. This language has statements that fall in one of 4 categories:

- Access Control Requirement (ACR): Traffic of the specified type from a network element A to a network element B should be allowed, or should be denied.
- Secure Content Requirement (SCR): Traffic of the specified type from A to B should be protected in a specific way (encrypted and/or authenticated, in tunnel or transport mode, with a specified cryptographic method).
- Content Access Requirement (CAR): Traffic of the specified type from A to B should be “readable” by (not encrypted at) network element C.

- Security Association Restriction (SAR): A security association or tunnel of the specified type may *not* be set up between points A and B.

This language we believe allows the expression of the most common security requirements in a natural way.

Given a security requirement, we need to determine if a specific security configuration (set of SAs) fulfills or satisfies that requirement. We formally defined necessary and sufficient conditions for satisfying each of the above types of requirements. We then developed an algorithm for determining whether a set of requirements is satisfied by a security configuration. This algorithm is based on path traversal in the network for each security requirement, examining the security policies or tunnels at each network element on the path between the two specified endpoints. By doing this for every path traversed by the specified traffic, it is possible to verify that the security requirement is satisfied along every such path.

When a configuration of security tunnels does *not* satisfy the high-level security requirements, it is desirable to modify the configuration to minimize the security violations or risk. Each violation of a security requirement is assigned a penalty, reflecting the severity of that violation. The task of security policy reconfiguration is to modify the existing tunnels to minimize the sum of the penalties (i.e., reduce to 0 if possible). Eliminating violations requires decomposing tunnels into shorter tunnels, subject to the security association restrictions. The process of finding a decomposition which minimizes the penalty we solve by brute force; enumeration of all decompositions, with pruning performed by a branch-and-bound algorithm to speed up the solution.

The final part of this part of the project addresses the issue of automatically generating, or synthesizing, a security configuration (set of SAs) that meets a specified security requirement or requirements. We investigated two approaches to accomplish this. The first approach, bundling, starts with a graph-based representation of the security requirements. From these requirements, the traffic flows are grouped into disjoint "bundles" sharing the same security requirements and sharing a common sub-path through the network. Then for each bundle, a filter specifying this group of flows is constructed, and the security tunnels or tunnels for this bundle is generated. The second approach, the direct approach, starts by synthesizing a tunnel for each SCR. Where tunnels overlap (share portions of the same path through the network), they are decomposed into a larger set of non-overlapping tunnels that fulfills all requirements.

We have done experiments with these two approaches on a large set of security requirements. While both approaches correctly find a solution, the direct approach is faster to compute, and results in a smaller number of SAs than the bundling approach, and is thus preferable.

Results and Discussion

Our work has shown how to easily specify a security requirement, how to verify that a set of IPSec security associations satisfies the requirement, and how to automatically generate a set of SAs from scratch, given a security requirement. We believe our methods will help eliminate the mistakes that occur during manual security tunnel configuration, due to complexity, lack of understanding of the interaction between multiple, independent security requirements, and lack of knowledge of the network topology.

The results from our research on this problem include the following:

- A paper and presentation by Zhi Fu, S. Felix Wu, et al. entitled "IPSec/VPN Security Policy: Correctness, Conflict Detection and Resolution", published in the *Proceedings of the Policy Workshop 2001*, Bristol, England, January 2001.

- A paper by Zhi Fu and S. Felix Wu entitled "Automatic Generation of IPSec/VPN Security Policies In an Intra-Domain Environment", was presented at and published in the *Proc. Of the12th International Workshop on Distributed Systems: Operations & Management (DSOM 2001),* Nancy France, Oct. 2001. This paper won a Best Paper Award at the conference.
- Software implementing the bundle and direct approaches for synthesizing IPSec security associations from a set of security requirements has been put on the website. The software is written in C for the Linux operating system. Source code is provided as well as the executable.

Zhi Fu's PhD thesis investigated in detail the problem and methods described in this chapter. The title was *Network Management And Intrusion Detection For Quality of Network Services*. She successfully defended her PhD thesis in Computer Science from NC State University in June of 2001, and is now employed as a staff researcher at Motorola Research Labs in Chicago, Illinois.

Conclusions and Recommendations

We believe our work will result in fewer mistakes in IPSec configuration, and make the task of generating an efficient set of security tunnels much easier. The net result is a higher degree of confidence in the correct implementation of a strong security posture. We are pursuing open problems in security policy specification and automatic configuration, and the possibility of standardizing the security requirements language we have defined. We hope to see products from network product vendors incorporating some of the capabilities described in this chapter.

References

[RFC2401] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol", IETF RFC 2401, Nov. 1998.

Chapter 8

Tracing Attack Traffic Through Stepping Stones Using Watermarking

Summary

Attackers who use the Internet to launch attacks often stage attacks through a series of unsuspecting (non-malicious) host computers, called stepping stones. They do this in order to make tracing the origin of the attacks difficult. Conventional IP Traceback attacks will only identify the last stepping stone in the sequence, but not the true origin of the attacks, or the other stepping stones.

We have developed a method of tracing attack traffic through stepping stones by correlating incoming traffic with outgoing traffic. We use both the packet payload and the packet timing characteristics for purposes of correlation. We have proposed an architecture for implementing our method, and demonstrated the method is highly resistant against common countermeasures the attacker may use to thwart tracing and correlation.

Introduction

Most attacks on computer systems, and theft of computer data, occurs via the Internet. The perpetrators of these attacks are anxious to hide their identity, so that they can (a) continue the attack and (b) be safe from prosecution or retaliation. There are many techniques for hiding the source and path of the attack, including the following:

1. Encrypting the traffic so the payload can't be recognized or analyzed
2. Mixing the attack traffic with other traffic to make it difficult to trace
3. Making the attack traffic look as innocuous and unremarkable as possible to avoid detection
4. Spoofing the source IP address (i.e., replacing the IP address of the true source of the attack traffic with the IP address of some other host)
5. Recruiting or infecting other hosts (i.e., implanting attack software on "zombies") to launch attacks on command from the attacker
6. Launching an attack through a series of unsuspecting / unwitting hosts ("stepping stones")
7. "laundering" the attack path by routing traffic through anonymizers (anonymizing proxies) or through networks with non-cooperative owners (i.e., belonging to nations not bound by treaty obligations) or by tunneling traffic between gateways

Much attention has been given recently to methods for IP traceback, whose goal is to identify the path that traffic takes through the Internet [SWKA2000]. Traceback is specifically intended to deal with concealment technique #4 (source address spoofing). Most of the methods assume that the routers in the network cooperate in the effort, and these methods specifically do not solve the problems of stepping stones or attack laundering (techniques #6 and #7). Both are serious limitations.

There have been several proposals for tracing traffic through stepping stones [ZP2000]. We first present our method and then describe its advantages over these other approaches. We aim for a method that:

- Is very effective (successfully traces attack traffic, and rarely identifies or follows a "false trail")
- Is effective despite the use of the above techniques, and is resistant to efforts by the attacker to defeat the specific tracing method
- Is difficult to detect when in operation
- Has low overhead (computation, bandwidth)
- Allows tracing in as little time as possible, since attacks may not last very long

Methods, Assumptions, Procedures

We have developed three methods of tracing: by (1) actively marking the payload, (2) passively monitoring the traffic timing characteristics, and (3) actively marking the traffic timing. All of our methods make use of the following framework or architecture.

We assume that at some point the attack traffic can be subtly modified to help in tracing. If the attack is interactive (i.e., the attacker telnets into a host), then the traffic being sent or echoed back to the attacker may be modified by the victim, who presumably has some incentive to cooperate. However, the point of modification may also be any routers or firewalls at any point of the network. We also assume the existence of observation points in the network to look for traffic meeting certain specified criteria. Lastly, we assume that communication between the modifier and the observation points is possible, to coordinate the process of tracing. A diagram of this architecture is shown in the figure below.

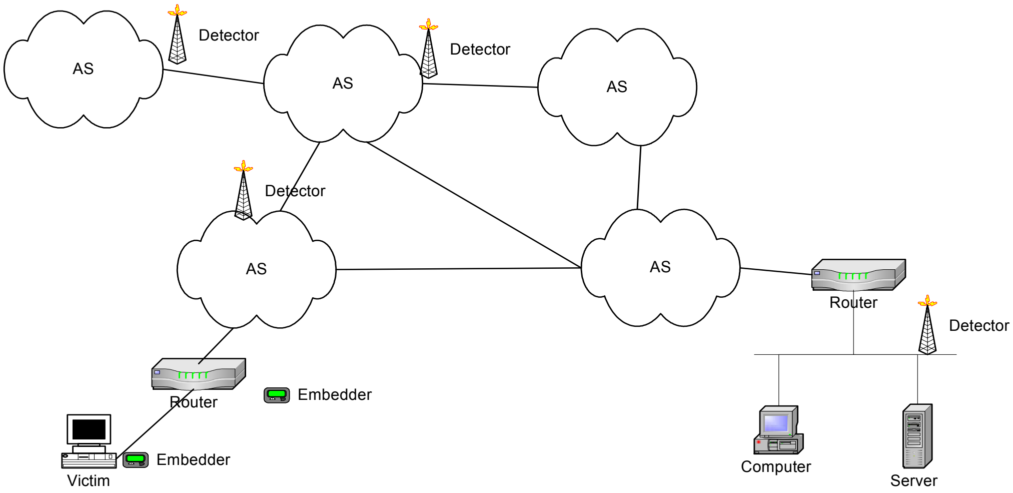


Figure 10. An example deployment of the tracing system

Method 1 is based on the principles of steganography, which is the science of concealing messages by embedding them in other information. We suggested that non-printing characters could be added to the payload of text when the attack consisted of conventional (typed) commands and feedback to be displayed on the attacker's screen. Since the attack payload is normally preserved through stepping stones, and embedded character sequences could be made arbitrarily long, the method was highly effective. However, it would not work if the attacker encrypted the attack traffic at any stepping stone, and it was relatively easy for the attacker to detect and remove.

Method 2 is a passive method which simply observes and records the timing characteristics of attack traffic. By "timing characteristics", we simply mean the intervals of time between successive packets in a TCP connection. We showed on a large set of traces that timing characteristics of traffic are highly distinctive, and that a tracing method based on timing could be extremely effective. As few as 25 or 30 packets were needed to uniquely identify almost every traffic source, and the timing characteristics were preserved even when the traffic traveled through scores of routers and many stepping stones. A figure of performance is shown below. In this figure, there should be 15 true positives, and there should be 0 false positives out of a total of 225 possible correlations. The limitation of this method, unfortunately, is that it is relatively easy for the attacker to modify the timing characteristics so they are no longer effective for tracing purposes. That is, at any stepping stone, the individual packets could be delayed specifically for the purpose of distorting the traffic timing to interfere with timing analysis.

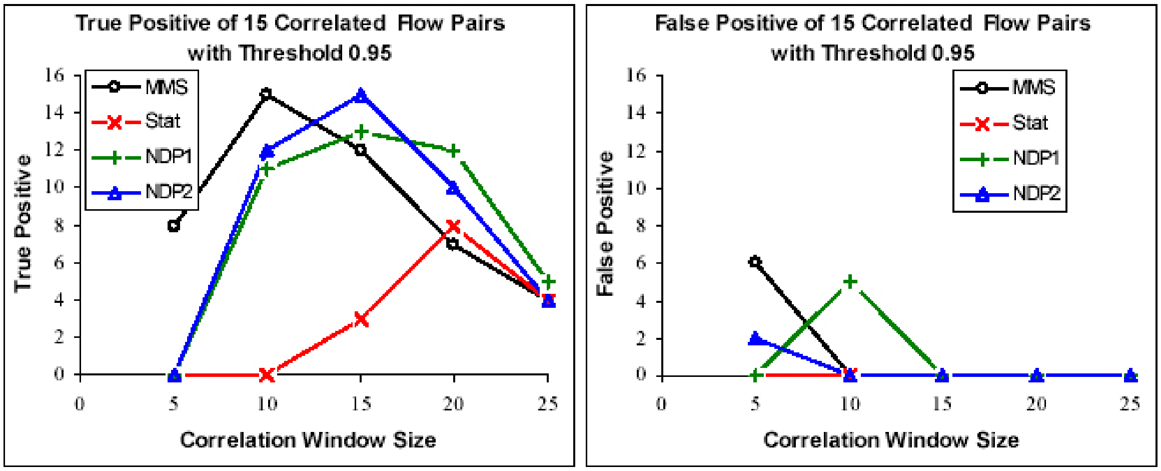


Figure 11. True and false positive rates on a test case, using passive timing analysis.

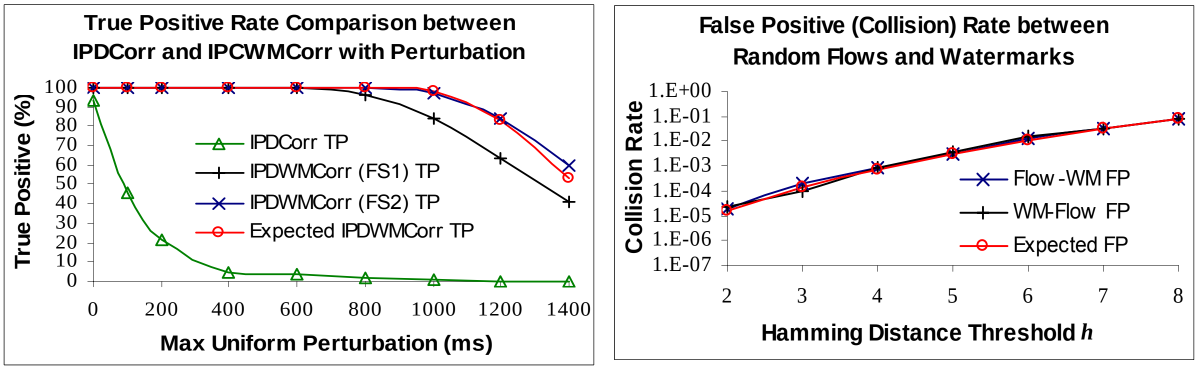
Method 3 is an active method, again based on the traffic timing. The intention of method 3 is to retain the advantages of method 2, but make it resistant to modification of the timing characteristics by the attacker. Our method perturbs packet timing in a way which is relatively easy to identify with specific knowledge, but difficult (for the attacker) to otherwise detect, and very difficult to remove. The embedded, or watermarking, of information by perturbing packet timing has 4 components:

- A single bit (0 or 1) of identifying information can be embedded by changing the interpacket delay of just two packets by a specified amount
- A single bit can be embedded more robustly by changing the *average* interpacket delay of a sequence of packets by a specified amount
- Multiple bits of identifying information can be embedded to improve the uniquely identifying characteristics of the embedded information
- A Hamming distance threshold allows identification of embedded information within a specified bound on the number of corrupted bits

The location of interpacket delays that are perturbed can be randomly generated in a way that makes detection by an attacker very difficult. The amount of perturbation of individual interpacket delays can be made arbitrarily small, while maintaining detectability in the presence of arbitrarily large timing modifications by the attacker.

Results and Discussion

We have run extensive experiments on large-scale traces of traffic from the Internet to validate our methods. The results for method #3 (active timing modification) are shown below. These results demonstrate the method is highly effective, and robust to timing modifications.



a. Correlation True Positive Rates under Random Timing Perturbs

b. Correlation False Positive (Collision) Rate vs Hamming Distance Threshold *h*

Figure 12. Results of watermarking flows using active timing modification.

Most previous work on tracing through stepping stones either is vulnerable to modifications of the payload (e.g., encryption), or is sensitive to attempts by the attacker to modify the packet flow timing characteristics. [D2002] suggests the usefulness of packet timing for tracing, but does not describe an active method for perturbing the timing to enhance traceability.

Outputs of this part of the project include the following:

- X. Wang, D. Reeves, F. Wu, and J. Yuill, "Sleepy Watermark Tracing: An Active Network-Based Intrusion Response Framework", *Proc. of IFIP 16th International Conference on Information Security*, Paris, France, June 2001.
- X. Wang, D. Reeves, and S. F. Wu, "Tracing Based Active Intrusion Response", *Journal of Information Warfare*, Teamlink Australia Pty Ltd., Vol. 1, No. 1, September 2001, pp. 50—61.
- X. Wang, D. Reeves, and S. F. Wu, "Inter-Packet Delay Based Correlation for Tracing Encrypted Connections Through Stepping Stones", *Proc. Of the 7th European Symposium on Research in Computer Security (ESORICS 2002), Springer-Verlag LNCS 2502.*
- X. Wang and D. S. Reeves will present their work at the ACM Conference on Computer and Communications Security (CCS 2003) in Washington, DC this month. The paper will be published as: X. Wang and D. S. Reeves, "Robust Correlation of Encrypted Attack Traffic Through Stepping Stones by Manipulation of Interpacket Delays", in the *Proc. Of ACM CCS 2003*, October 2003.

In addition, the work described in this section is the basis for the PhD thesis of Xinyuan Wang, who expects to defend his PhD thesis in December of 2003. He is currently employed by Cisco Networks.

Conclusions and Recommendations

Tracing of attack traffic is needed to prosecute and isolate the source of attacks. Current methods are fairly easy for attackers to defeat. We have developed a method which is potentially much more robust against a variety of countermeasures (#1, #3, #4, and #6 above). We will continue to develop and promote this new method.

The countermeasures that have not been addressed yet by us are anonymizing proxies and traffic padding, or "camouflaging". We will shortly turn our attention to these issues.

References

[SWKA2000] S. Savage, D. Wetherall, A. Karlin and T. Anderson. (2000) Practical Network Support for IP Traceback. Proceedings of the ACM SIGCOMM '2000.

[ZP2000] Y. Zhang and V. Paxson. "Detecting Stepping Stones", In *Proceedings of 9th USENIX Security Symposium*, 2000.

[D2002] D. Donoho et al., "Multiscale Stepping Stone Detection: Detecting Pairs of Jittered Interactive Streams by Exploiting Maximum Tolerable Delay", In *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002)*, October, 2002. Springer Verlag Lecture Notes in Computer Science, #2516.